

## Laboratorio Nro. 4

### Árboles binarios

### Objetivos

1. Utilizar árboles binarios para resolver problemas
2. Entender la eficiencia de los árboles binarios de búsqueda

### Consideraciones Iniciales

#### Leer la Guía



Antes de comenzar a resolver el presente laboratorio, leer la **“Guía Metodológica para la realización y entrega de laboratorios de Estructura de Datos y Algoritmos”** que les orientará sobre los requisitos de entrega para este y todos los laboratorios, las rúbricas de calificación, el desarrollo de procedimientos, entre otros aspectos importantes.

#### Registrar Reclamos



En caso de tener **algún comentario** sobre la nota recibida en este u otro laboratorio, pueden **enviarlo** a través de <http://bit.ly/2q4TTKf>, el cual será atendido en la menor brevedad posible.

**Traducción de Ejercicios** En el GitHub del docente, encontrarán la traducción al español de los enunciados de los Ejercicios en Línea.



**Visualización de  
Calificaciones**



A través de **Eafit Interactiva** encontrarán **un enlace** que les permitirá **ver un registro de las calificaciones** que **emite el docente** para cada taller de laboratorio y según las rubricas expuestas. **Véase sección 3, numeral 3.7.**

**GitHub**

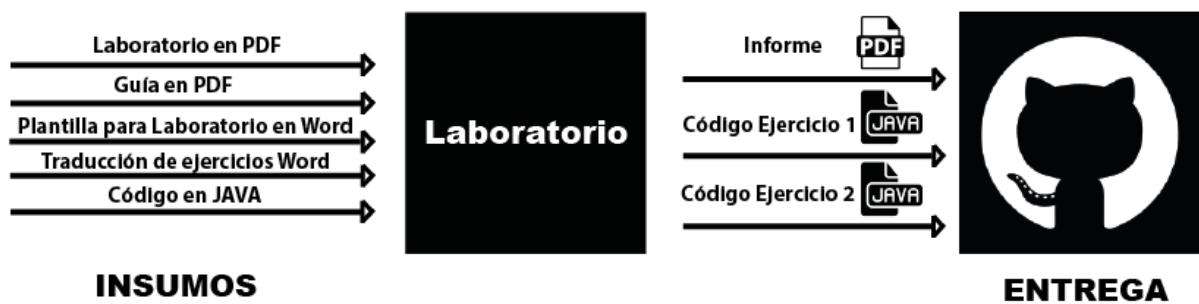


Crear un repositorio en su cuenta de GitHub con el nombre `st0245-suCodigoAqui`. **2.** Crear una carpeta dentro de ese repositorio con el nombre `laboratorios`. **3.** Dentro de la carpeta `laboratorios`, crear una carpeta con nombre `lab05`. **4.** Dentro de la carpeta `lab05`, crear tres carpetas: `informe`, `codigo` y `ejercicioEnLinea`. **5.** Subir el informe pdf a la carpeta `infome`, el código del ejercicio 1 a la carpeta `codigo` y el código del ejercicio en línea a la carpeta `ejercicioEnLinea`. Así:

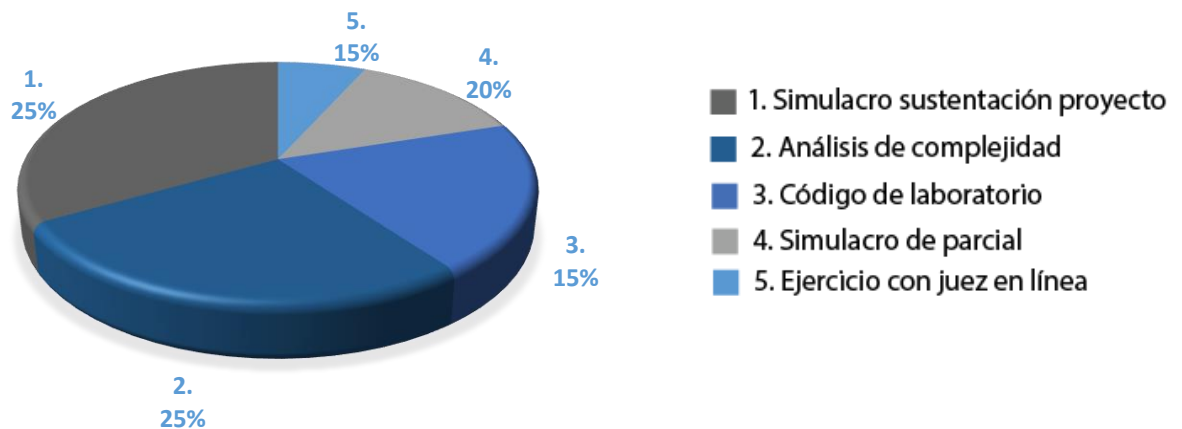
```
st0245-suCodigoAqui
  laboratorios
    lab01
      informe
      codigo
      ejercicioEnLinea
    lab02
    ...
```

## Intercambio de archivos

Los archivos que **ustedes deben entregar** al docente son: **un archivo PDF** con el informe de laboratorio usando la plantilla definida, y **dos códigos**, uno con la solución al numeral 1 y otro al numeral 2 del presente. Todo lo anterior se entrega en **GitHub**.



## Porcentajes y criterios de evaluación para el laboratorio



DOCENTE MAURICIO TORO BERMÚDEZ  
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)

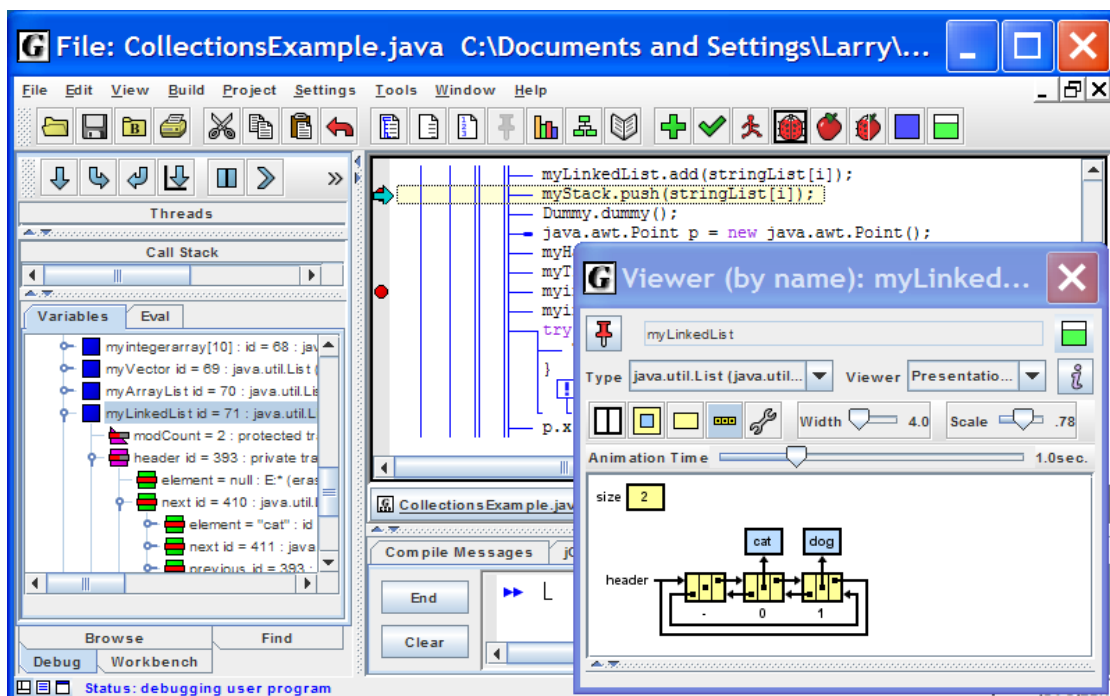
## Resolver Ejercicios



**NOTA 1:** Para este taller, de manera especial, se recomienda utilizar el *IDE Jgrasp* (<http://www.jgrasp.org/>) porque tiene un depurador gráfico excelente para estructuras de datos



**NOTA 2:** Véase a continuación gráfica de Jgrasp para efectos de ejemplificación



## 1. Códigos para entregar en GitHub:



En la vida real, la documentación del software hace parte de muchos estándares de calidad como CMMI e ISO/IEC 9126



Véase Guía **en Sección 3, numeral 3.4**



Código de laboratorio en **GitHub**. Véase Guía en **Sección 4, numeral 4.24**

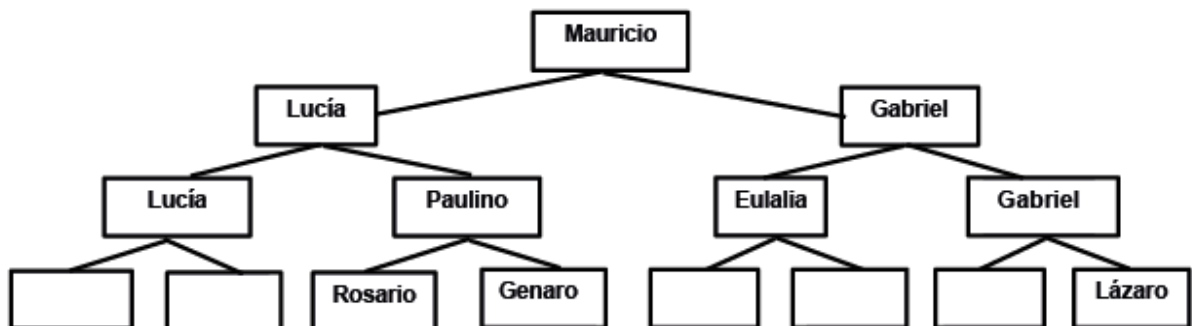


*Entregar documentación en **JavaDoc** o equivalente. El uso de JavaDoc es opcional*



*No se reciben archivos en **.RAR** ni en **.ZIP***

**1.1** Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos. Con esa información, construyan sus árboles genealógicos en Java usando la clase `BinaryTree`. Como un ejemplo, este sería el mío:



DOCENTE MAURICIO TORO BERMÚDEZ  
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627  
Correo: [mtorobe@eafit.edu.co](mailto:mtorobe@eafit.edu.co)



En la vida real, los árboles de ancestros son utilizados en Filogenética. Filogenética es la parte de la biología que se ocupa de determinar las relaciones evolutivas entre diferentes grupos de organismos

### 1.2 Teniendo en cuenta lo anterior, modelen sus ascendentes directos como un árbol binario utilizando la implementación en *Java* de árboles binarios.

Si no conocen el nombre de uno de ellos, no lo escriban. No modelen hermanos ni tíos. Modelen en el método `main`, como hijos derechos, los hombres y como hijos izquierdos, las mujeres.

### 1.3 Utilicen las funciones *buscar*, *tamaño*, *altura* e *imprimir recursivamente* para verificar que el árbol quedó construido correctamente. Demuéstrenlo en el `main`.

1.4 Wilkenson recolectó el árbol familiar de su familia con mucho cariño para dárselo a su abuela Eustaquia. Wilkenson, el nieto de Eustaquia, lo ingresó a *Java* usando las clases `Node` y `BinaryTree`.

**Escriban un método para la clase `BinaryTree` que calcule quién es la abuela materna de una persona. No escriban el método `main`.**









**NOTA 1:** El método debe funcionar para cualquier árbol genealógico, no solo para el Wilkenson. Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la abuela materna



**NOTA 2:** Considere que los ancestros paternos van a la derecha y los ancestros maternos a la izquierda.

## 2) Ejercicios en línea sin documentación en GitHub

|   |   |   |   |
|---|---|---|---|
|  | Véase Guía en <b>Sección 3, numeral 3.3</b>                               |  | <b>No entregar</b> documentación <b>HTML</b>  |
|  | Entregar un archivo en <b>.JAVA</b>                                       |  | <b>No se reciben</b> archivos en <b>.PDF</b>  |
|  | <b>Resolver</b> los problemas de <b>CodingBat</b> usando <b>Recursión</b> |  | Código del ejercicio en línea en <b>GitHub</b> . Véase Guía en <b>Sección 4, numeral 4.24</b> |

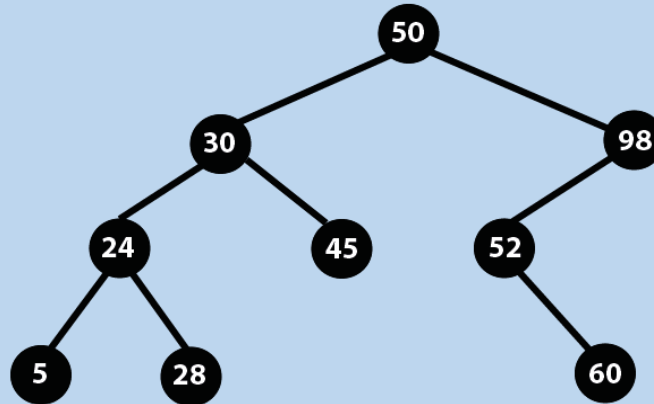


**NOTA 1:** Recuerden que, **si toman la respuesta de alguna fuente**, deben **referenciar** según el **tipo de cita** correspondiente. Véase *Guía en Sección 4, numerales 4.16 y 4.17*

### 2.1 Resuelvan el siguiente problema usando árboles binarios

Un árbol de búsqueda binario (o árbol binario de búsqueda) es un árbol binario que satisface las siguientes propiedades:

1. El subárbol izquierdo de un nodo contiene sólo nodos con valores menores al valor del nodo.
2. El subárbol derecho de un nodo contiene sólo nodos con valores mayores al valor del nodo.
3. Tanto el subárbol izquierdo como el subárbol derecho tienen que ser también árboles de búsqueda binarios.



**Figura 1. Ejemplo de un árbol de búsqueda binario**

El recorrido en pre-orden (Raíz – Izquierda - Derecha) imprime los nodos visitando la raíz del árbol, luego recorriendo el subárbol izquierdo y luego recorriendo el subárbol derecho.

El recorrido en pos-orden (Izquierda – Derecha - Raíz) imprime el subárbol izquierdo primero, luego el subárbol derecho y por último la raíz. Por ejemplo, los resultados del recorrido en pre-orden y pos-orden del árbol de búsqueda binario mostrado en la Figura 1 son:

- a) Pre-orden:** 50 30 24 5 28 45 98 52 60
- b) Pos-orden:** 5 28 24 45 30 60 52 98 50

Dado el recorrido en pre-orden de un árbol de búsqueda binario, su tarea es encontrar el recorrido en pos-orden del árbol.

### **Entrada**

El valor de cada nodo del árbol de búsqueda binario se dará de acuerdo a un recorrido en pre-orden. Cada nodo tiene un valor que es un entero positivo menor que  $10^6$ . Todos los valores se dan en líneas separadas (un entero por línea). Ud. puede asumir que un árbol de búsqueda binario no contiene más de 10 000 nodos y que no hay nodos



duplicados (es decir, el valor de cada nodo es único en el árbol de búsqueda binario dado).

### Salida

La salida contiene el resultado del recorrido en pos-orden del árbol de búsqueda binario dado en la entrada. Impriman un valor por línea.

### Ejemplo de entrada


50  
30  
24  
5  
28

45  
98  
52  
60

### Ejemplo de salida

5  
28  
24  
45  
30  
60  
52  
98  
50

**2.2 [Ejercicio Opcional]** Resuelvan el siguiente problema <http://bit.ly/2iNuHaa>

|   |   |   |
|---|---|---|
|  | <p>UNIVERSIDAD EAFIT<br/>ESCUELA DE INGENIERÍA<br/>DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS</p> | <p>Cód. ST0245<br/>Estructuras de Datos 1</p> |
|---|---|---|

### 3) Simulacro de preguntas de sustentación de Proyectos

|  |   |  |   |
|--|---|--|---|
|  | Véase Guía en <b>Sección 3, Numeral 3.5</b>                 |  | <i>Entregar informe de laboratorio en PDF</i> |
|  | <i>Usen la <b>plantilla</b> para responder laboratorios</i> |  | <b>No apliquen Normas Icontec</b> para esto   |

**3.1** ¿Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?

**3.2** Expliquen con sus propias palabras cómo funcionan los ejercicios de los numerales 2.1 y 2.2

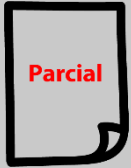


**NOTA 1:** Recuerden que debe explicar su implementación en el informe PDF

**3.3** Calculen la complejidad de los ejercicios realizados en los numerales 2.1 y 2.2 y agréguenla al informe PDF

**3.4** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.4

#### 4) Simulacro de Parcial en el informe PDF



Para este simulacro, agreguen ***sus respuestas*** en el informe PDF.



*El día del Parcial no tendrán computador, JAVA o acceso a internet.*

##### 1. El nodo de un árbol binario se define así:

```
class Nodo {  
    Nodo izq;  
    Nodo der;  
    int dato;  
}
```

Kefo ha dañado a Dayla su código para determinar **cuál es la altura máxima de un árbol binario**. Dayla no recuerda como lo había hecho y les pide ayuda para que lo completen.

```
01 int altura(Nodo raiz){  
02     if(raiz == null)  
03         return 0;  
04     int izq = _____;  
05     int der = _____;  
06     return Math.max(izq, der);}
```

##### a) Completen el espacio en línea 04 (10%)

\_\_\_\_\_

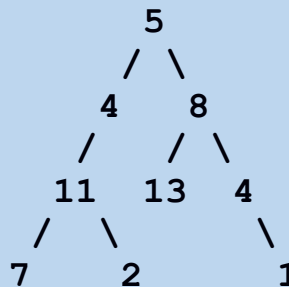
**b) Completen el espacio en línea 05 (10%)**

**2.** A un árbol binario de búsqueda se le ingresan 9 elementos en este orden: 12, 3, 5, 8, 2, 1, 9, 17. **¿Cuántos nodos hay que recorrer antes de encontrar el número 8?** Se cuenta la raíz, pero no se cuenta el nodo donde está el 8.

- a) 2
- b) 1
- c) 3
- d) 5

**3.** Definimos el camino desde la raíz hasta una hoja en un árbol binario como una secuencia de nodos empezando en el nodo raíz y bajando hasta una hoja. Una hoja es un nodo que no tiene hijos. Decimos que un árbol vacío no contiene caminos desde la raíz hasta una hoja.

Como un ejemplo, el siguiente árbol tiene 4 caminos desde la raíz hasta una hoja:



Caminos desde la raíz hasta una hoja:

- ☒ **Camino 1:** 5 4 11 7
- ☒ **Camino 2:** 5 4 11 2
- ☒ **Camino 3:** 5 8 13
- ☒ **Camino 4:** 5 8 4 1

Para este problema nos interesan las sumas de los elementos en los nodos de esos caminos, por ejemplo, la suma del camino [5,4,11,7] es 27.

Dada la raíz de un árbol binario `Nodo a` y un entero `int suma`, **decir si existe un camino desde la raíz hasta una hoja tal que al sumar los valores de los nodos de ese camino la suma sea igual al parámetro `suma`.**

Retorne falso si no se puede encontrar un camino con esa condición. Utilice la definición de `Nodo` del punto 1.

Desafortunadamente, al código que hizo Dayla le faltan unas líneas y Kefo está de vacaciones.

```
1 boolean sumaElCamino(Nodo a, int suma) {  
2   if (a == null)  
3     return _____;  
4   if (a.izq == null && a.der == null)  
5     return suma == _____;  
6   else  
7     return sumaElCamino(_____, _____)  
8     || sumaElCamino(_____, _____); }
```

**a) Completen el espacio de la línea 03 (10 %)**

\_\_\_\_\_

**b) Completen el espacio de la línea 05 (10 %)**

\_\_\_\_\_

**c) Complete los espacios de la línea 07 (10 %)**

\_\_\_\_\_, \_\_\_\_\_

**d) Complete los espacios de la línea 08 (10 %)**

\_\_\_\_\_, \_\_\_\_\_

**4.** Consideren la siguiente definición de árbol binario:

```
class Node {  
    public Node left;  
    public Node right;  
    public String data;  
    public Node(String d) {  
        data = d;  
    }  
}
```

El siguiente algoritmo imprime todos los valores de un árbol en pre orden.

```
01 private void printAUX(Node node) {  
02     if (node != null) {  
03         System.out.println(node.data);  
04         printAUX(node.left);  
05         printAUX(node.right);  
06     }  
07 }  
08 public boolean print() {  
09     printAUX(root);  
10 }
```

**4.1** ¿Cuál ecuación de recurrencia que describe el número de instrucciones que ejecuta el algoritmo print en el peor de los casos?

La variable  $n$  representa el número de elementos del árbol.

- a)  $T(n) = T(n-1) + C$
- b)  $T(n) = 2 \cdot T(n-1) + C$
- c)  $T(n) = 2 \cdot T(n/2) + C$
- d)  $T(n) = T(n/2) + C$
- e)  $T(n) = T(n+1) + C$

**4.2** ¿Cuál es su complejidad asintótica en el peor de los casos del algoritmo print? La variable  $n$  representa el número de elementos del árbol.

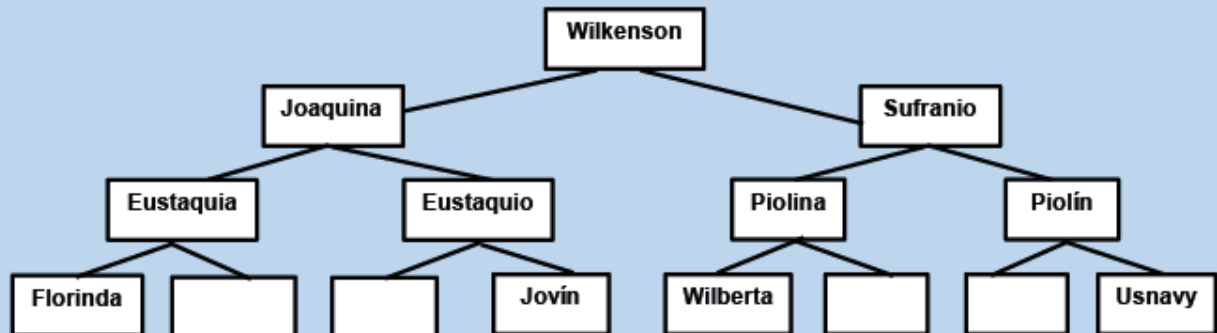
- a)  $O(n)$
- b)  $O(n^2)$
- c)  $(\log n)$
- d)  $O(n.m)$
- e)  $O(1)$

**4.3** ¿Cuál es la salida del algoritmo print para el siguiente árbol?

Tenga en cuenta que la raíz es Wilkenson.

Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha.

Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.



- a) Wilkenson, Sufranio, Piolín, Usnavy, Piolina, Wilberta, Joaquina, Eustaquio, Florinda, Eustaquia, Yovín
- b) Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda, Wilkenson, Yovín, Eustaquio
- c) Wilkenson, Yovín, Eustaquio, Sufranio, Piolina, Wilberta, Piolín, Usnavy, Joaquina, Estaquia, Florinda

- d) Wilkenson, Joaquina, Eustaquia, Florinda, Eustaquio, Jovín, Sufranio, Piolina, Wilberta, Piolín, Usnavy
- e) Sufranio, Piolina, Wilberta, Piolín, Usnavy, Florinda, Wilkenson, Yovín, Eustaquio, Joaquina, Estaquia

**4.4** ¿Qué modificación hay que hacer algoritmo print para que arroje la siguiente respuesta para el árbol anterior?:

Usnavy, Piolín, Wilberta, Piolina, Sufranio, Florinda, Eustaquio, Yovín, Eustaquia, Joaquina, Wilkenson.

Tenga en cuenta que la raíz es Wilkenson. Como Wilkenson es la raíz, para poder entender el árbol, recomendamos rotarlo 180 grados.

Después de hacer la rotación, usted entenderá que realmente las mujeres van a la izquierda y los hombres a la derecha.

Tenga en cuenta que, en un árbol genealógico, para algunas personas, no se conoce la mamá o el papá.

- a) Cambiar el orden de las líneas 03, 04 y 05 por 05, 04, 03
- b) Cambiar el orden de las líneas 03, 04 y 05 por 04, 05, 03
- c) Cambiar el orden de las líneas 03, 04 y 05 por 03, 05, 04
- d) Cambiar el orden de las líneas 03, 04 por 06, 07
- e) Intercambiar la línea 03 y la línea 04

**5.** Luis escribió un programa para insertar un número en un árbol binario de búsqueda. En dicho árbol, él quiere tener los números menores o iguales a la raíz a la derecha y los mayores a la raíz a la izquierda.

Ayúdele a completar su código. El algoritmo recibe la raíz de un árbol `p` y un número `a` insertar `toInsert`, y retorna la raíz del árbol con el elemento insertado donde corresponde.

```
01 private Node insert(Node p,int toInsert){
02   if (p == null)
03     return new Node(toInsert);
```



```
04  if (.....)
05    return p;
06  if (.....)
07    p.left = insert(p.left, toInsert);
08  else
09    p.right = insert(p.right, toInsert);
10  return p;
11 }
```

**a)** Complete, por favor, la línea 4 con la condición que corresponde **(10 %)**

.....

**b)** Complete, por favor, la línea 6 con la condición que corresponde **(10 %)**

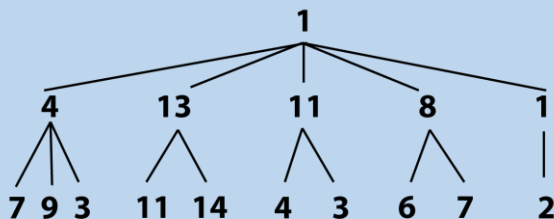
.....

**6.** Dado un árbol *n-ario*, un camino desde la raíz a cualquiera de sus hojas se considera **simple** si la suma de sus elementos pares es igual a la suma de sus elementos impares.

Por ejemplo, un camino desde la raíz hasta una hoja con los elementos [1, 2, 1] es **simple**, pero el camino desde la raíz hasta una hoja [1, 3, 1] **no es simple**.

Su tarea es determinar cuántos caminos **simples** hay en un árbol. La implementación de un nodo *n-ario* es la siguiente:

```
class NNode{
    int val; //Valor en el nodo actual.
    //Hijos del nodo actual.
    LinkedList<NNode> hijos;
}
```



**6.1** ¿Cuántos caminos **simples** hay desde una raíz hasta una hoja en el árbol anterior?

- a. 1
- b. 2
- c. 3
- d. 4

El siguiente código nos ayuda a determinar el número de caminos simples en un árbol, pero faltan algunas líneas. Por favor, complétenlas.

```
01 public int cuantosSimples(NNodo raiz, int suma){
02     //Arbol vacio
03     if(raiz == null)
04         _____;
05     //Hoja
06     if(raiz.hijos.size() _____) // Si suma es 0
07         return (suma == 0) ? 1 : 0; // Retorne 1, sino, 0
08     int total = 0;
09     for(NNodo n: raiz.hijos)
10         if(n.val % 2 == 0) // Par
11             total += cuantosSimples(n, suma + n.val);
12         else // Impar
13             total += cuantosSimples(n, suma - n.val);
15     return total;
16 }
17
18 public int cuantosSimples(NNodo raiz){
19     int val = (raiz.val % 2 == 0) ?
20         raiz.val : -raiz.val;
21     return cuantosSimples(raiz, val);
22 }
```

**6.2** Completen la línea 4

\_\_\_\_\_

### 6.3 Completen la línea 6

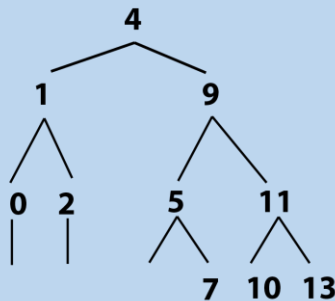
---

7. Los siguientes algoritmos son el recorrido en in orden y pos orden. En la vida real, estos recorridos son de utilidad para hacer procesamiento del lenguaje natural.

```
void InOrden(Node node) {  
    if (node != null) {  
        InOrden(node.left);  
        System.out.println(node.data);  
        InOrden(node.right);  
    }  
}
```

```
void PosOrden(Node node) {  
    if (node != null) {  
        PosOrden(node.left);  
        PosOrden(node.right);  
        System.out.println(node.data);  
    }  
}
```

Consideren el recorrido in orden y pos orden de un **Árbol binario de búsqueda** con los siguientes elementos 4, 9, 1, 5, 7, 11, 13, 2, 0, 10. Ahora, la impresión del recorrido in-orden nos devolverá los elementos , y, la impresión del recorrido pos-orden devolverá .



7.1 ¿Cuál es la impresión pos-orden?

1. 0, 2, 1, 7, 5, 10, 13, 11, 9, 4
2. 0, 1, 2, 4, 5, 7, 9, 10, 11, 13
3. 0, 2, 1, 7, 10, 5, 13, 11, 9, 4
4. 4, 1, 0, 2, 9, 5, 7, 11, 13, 10

**7.2** Supongan que hacemos la comparación de 'y' , ¿cuál es la cantidad de elementos para los cuales no es igual 'a' ?

1. 7
2. 2
3. 5
4. 8

## 5. [Ejercicio Opcional] Lectura recomendada



"Quienes se preparan para el ejercicio de una profesión requieren la adquisición de competencias que necesariamente se sustentan en procesos comunicativos. Así cuando se entrevista a un ingeniero recién egresado para un empleo, una buena parte de sus posibilidades radica en su capacidad de comunicación; pero se ha observado que esta es una de sus principales debilidades..."

Tomado de <http://bit.ly/2qJKzJD>



Véase Guía en **Sección 3, numeral 3.5 y 4.20** de la Guía Metodológica, "Lectura recomendada" y "Ejemplo para realización de actividades de las Lecturas Recomendadas", respectivamente

Posterior a la lectura del "**Robert Lafore, Data Structures and Algorithms in Java (2<sup>nd</sup> edition), 2002. Chapter 8: Binary Trees.**", realicen las siguientes actividades que les permitirán sumar puntos adicionales:

- a) Escriban un resumen de la lectura que tenga una longitud de 100 a 150 palabras
- b) Hagan un mapa conceptual que destaque los principales elementos teóricos.



**NOTA 1:** Si desean una lectura adicional en inglés, consideren la siguiente: "**Narasimha Karumanchi, Data Structures and Algorithms made easy in Java, (2<sup>nd</sup> edition), 2011. Section 6.9 BST**", que encuentran en biblioteca.



**NOTA 2:** Estas respuestas también deben incluirlas en el informe PDF

## 6. [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual



El trabajo en equipo es una exigencia actual del mercado. "Mientras algunos medios retratan la programación como un trabajo solitario, la realidad es que requiere de mucha comunicación y trabajo con otros. Si trabajas para una compañía, serás parte de un equipo de desarrollo y esperarán que te comuniques y trabajes bien con otras personas"

Tomado de <http://bit.ly/1B6hUDp>



Véase Guía en **Sección 3, numeral 3.6** y **Sección 4, numerales 4.21, 4.22 y 4.23** de la Guía Metodológica

- a) Entreguen copia de todas las actas de reunión usando el tablero Kanban, con fecha, hora e integrantes que participaron
- b) Entreguen el reporte de *git*, *svn* o *mercurial* con los cambios en el código y quién hizo cada cambio, con fecha, hora e integrantes que participaron
- c) Entreguen el reporte de cambios del informe de laboratorio que se genera *Google docs* o herramientas similares



**NOTA 1:** Estas respuestas también deben incluirlas en el informe PDF

## 7. [Ejercicio Opcional] Laboratorio en inglés:



El inglés es un idioma muy importante en la Ingeniería de Sistemas porque la mayoría de los avances en tecnología se publican en este idioma y la traducción, usualmente se demora un tiempo y es sólo un resumen de la información original.

Adicionalmente, dominar el inglés permite conseguir trabajos en el exterior que son muy bien remunerados

Tomado de [goo.gl/4s3LmZ](https://goo.gl/4s3LmZ)

Entreguen el código y el informe traducido al inglés. Utilicen la plantilla dispuesta en este idioma para el laboratorio

## Resumen de Ejercicios a Resolver

**1.1** Investiguen cuáles son sus nombres, los nombres de sus padres, los nombres de sus abuelos y los nombres de sus bisabuelos. Con esa información, construyan sus árboles genealógicos en Java usando la clase `BinaryTree`

**1.2** Modelen sus ascendentes directos como un árbol binario utilizando la implementación en Java de árboles binarios.

**1.3** Utilicen las funciones *buscar*, *tamaño*, *altura* e *imprimir recursivamente* para verificar que el árbol quedó construido correctamente. Demuéstrenlo en el `main`.

**1.4** Escriban un método para la clase `BinaryTree` que calcule quién es la abuela materna de una persona. No escriban el método `main`.

**2.1** Resuelvan el siguiente problema usando árboles binarios

**2.2 [Ejercicio Opcional]** Resuelvan el siguiente problema <http://bit.ly/2iNuHaa>

**3.1** ¿Se puede implementar más eficientemente un árbol genealógico para que la búsqueda e inserción se puedan hacer en tiempo logarítmico? ¿O no se puede? ¿Por qué?

**3.2** Expliquen con sus propias palabras cómo funcionan los ejercicios de los numerales 2.1 y 2.2

**3.3** Calculen la complejidad de los ejercicios realizados en los numerales 2.1 y 2.2 y agréguela al informe PDF

**3.4** Expliquen con sus palabras las variables (qué es 'n', qué es 'm', etc.) del cálculo de complejidad del numeral 3.4

**4.** Simulacro parcial

**5.** [Ejercicio Opcional] Lectura recomendada

**6.** [Ejercicio Opcional] Trabajo en Equipo y Progreso Gradual

**7.** [Ejercicio Opcional] Laboratorio en inglés



## Ayudas para resolver los ejercicios

|                                   |                                |
|-----------------------------------|--------------------------------|
| Ayudas para el Ejercicio 1.....   | <a href="#"><u>Pág. 26</u></a> |
| Ayudas para el Ejercicio 1.3..... | <a href="#"><u>Pág. 26</u></a> |
| Ayudas para el Ejercicio 2.1..... | <a href="#"><u>Pág. 26</u></a> |
| Ayudas para el Ejercicio 2.2..... | <a href="#"><u>Pág. 27</u></a> |
| Ayudas para el Ejercicio 3.2..... | <a href="#"><u>Pág. 27</u></a> |
| Ayudas para el Ejercicio 3.4..... | <a href="#"><u>Pág. 27</u></a> |
| Ayudas para el Ejercicio 4.....   | <a href="#"><u>Pág. 27</u></a> |
| Ayudas para el Ejercicio 5a.....  | <a href="#"><u>Pág. 28</u></a> |
| Ayudas para el Ejercicio 5b.....  | <a href="#"><u>Pág. 28</u></a> |
| Ayudas para el Ejercicio 6a.....  | <a href="#"><u>Pág. 28</u></a> |
| Ayudas para el Ejercicio 6b.....  | <a href="#"><u>Pág. 28</u></a> |
| Ayudas para el Ejercicio 6c.....  | <a href="#"><u>Pág. 28</u></a> |

## Ayudas para el Ejercicio 1



**NOTA3:** Si deciden hacer la documentación de los puntos del numeral 1, vean la *Guía en Sección 4, numeral 4.1 “Cómo escribir la documentación HTML de un código usando Javadoc”*

## Ayudas para el Ejercicio 1.3



**PISTA 1:** Como un ejemplo, en el árbol genealógico de Wilkenson, Eva es la abuela materna de Wilkenson. Otro ejemplo, Wilberta es la abuela materna de Sufranio y no se conoce la abuela materna de Joaquina.

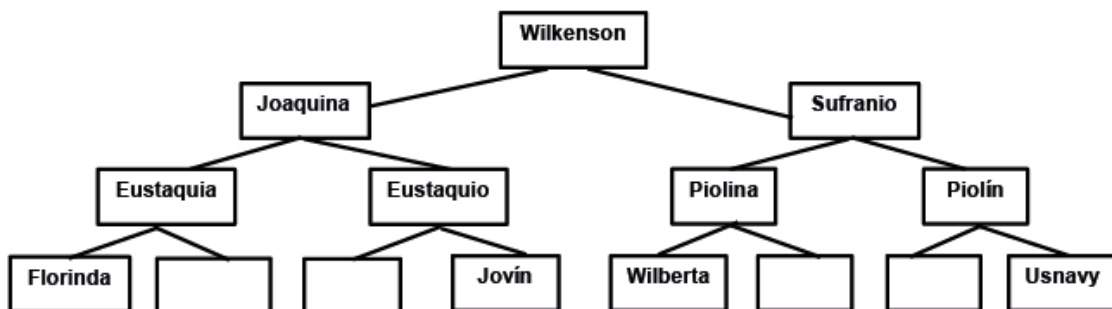


**PISTA 2:** Utilicen el siguiente estereotipo para el método:

```
public String getGrandMothersName(String name)
```



**PISTA 3:** Retornen una cadena vacía si no se encuentra el nombre de la persona *name* o la abuela materna de *name*. A continuación, un ejemplo de un árbol genealógico:



## Ayudas para el Ejercicio 2.1



**PISTA 1:** Léase sobre <http://bit.ly/1V5RqKL>

## Ayudas para el Ejercicio 2.2



**PISTA 1:** Use *DFS*. Es similar a lo que hacen en Lenguajes de Programación para construir el árbol sintáctico en una expresión aritmética con paréntesis.

## Ayudas para el Ejercicio 3.2



**PISTA 1:** Léase <http://bit.ly/1V5RqKL>



**PISTA 2:** Léase <http://bit.ly/2i8DgM4>

## Ayudas para el Ejercicio 3.4



**PISTA 1:** Véase *Guía en Sección 4, numeral 4.11 “Cómo escribir la complejidad de un ejercicio en línea”*



**PISTA 2:** Véase *Guía en Sección 4, numeral 4.19 “Ejemplos para calcular la complejidad de un ejercicio de CodingBat”*

## Ayudas para el Ejercicio 4



**PISTA 1:** Véase *Guía en Sección 4, Numeral 4.18 “Respuestas del Quiz”*



**PISTA 2:** Lean las diapositivas tituladas “*Data Structures I: Binary trees*”, encontrarán la mayoría de las respuestas

## Ayudas para el Ejercicio 5a



**PISTA 1:** En el siguiente enlace, unos consejos de cómo hacer un buen resumen <http://bit.ly/2knU3Pv>



**PISTA 2:** [Aquí](#) le explican cómo contar el número de palabras en Microsoft Word

## Ayudas para el Ejercicio 5b



**PISTA 1:** Para que hagan el mapa conceptual se recomiendan herramientas como las que encuentran en <https://cacoo.com/> o <https://www.mindmup.com/#m:new-a-1437527273469>

## Ayudas para el Ejercicio 6a



**PISTA 1:** Véase *Guía en Sección 4, Numeral 4.21 “Ejemplo de cómo hacer actas de trabajo en equipo usando Tablero Kanban”*

## Ayudas para el Ejercicio 6b



**PISTA 1:** Véase *Guía en Sección 4, Numeral 4.23 “Cómo generar el historial de cambios en el código de un repositorio que está en svn”*

## Ayudas para el Ejercicio 6c



**PISTA 1:** Véase *Guía en Sección 4, Numeral 4.22 “Cómo ver el historial de revisión de un archivo en Google Docs”*